Randomization

Robert C. Seacord, Software Engineering Institute [vita[1]]

2005-09-27

Randomization works on the principle that it is harder to hit a moving target. Addresses of memory allocated by `malloc()` are fairly predictable. Randomizing the addresses of blocks of memory returned by the memory manager can make it more difficult to exploit a heap-based vulnerability.

Development Context

Dynamic memory management

Technology Context

C++, C, UNIX, Win32

Attacks

Attacker executes arbitrary code on machine with permissions of compromised process or changes the behavior of the program.

Risk

Standard C dynamic memory management functions such as `malloc()`, `calloc()`, and `free()` [ISO/IEC 99] are prone to programmer mistakes that can lead to vulnerabilities resulting from buffer overflow in the heap, writing to already freed memory, and freeing the same memory multiple times (e.g., double-free vulnerabilities).

Description

Randomization works on the principle that it is harder to hit a moving target. Addresses of memory allocated by `malloc()` are fairly predictable. Randomizing the addresses of blocks of memory returned by the memory manager can make it more difficult to exploit a heap-based vulnerability.

Randomizing memory addresses can occur in multiple locations. For both the Windows and UNIX operating systems, the memory manager requests memory pages from the operating system, which are then broken up into small chunks and managed as required by the application process. It is possible to randomize both the pages returned by the operating system and the addresses of chunks returned by the memory manager.

The OpenBSD kernel, for example, uses `mmap()` to allocate or map additional memory pages. The `mmap()` function will return a random address each time an allocation is performed, as long as the MAP_FIXED flag is not specified. The `malloc()` function can also be configured to return random

1. daisy:274 (Seacord, Robert C.)

chunks.

The result is that each time a program is run, it exhibits different address space behavior, thereby making it harder for an attacker to guess the location of memory structures that must be overwritten to exploit a vulnerability.

Because randomization can make debugging difficult, it can usually be enabled or disabled at runtime. Also, randomization adds an unpredictable, but often significant, performance overhead.

References

[ISO/IEC 99]	ISO/IEC. *ISO/IEC 9899 Second edition 1999-12-01 Programming Languages — C*. International Organization for Standardization, 1999.

Pearson Education, Inc. Copyright

Fields

Name	Value
Copyright Holder	Pearson Education

Fields

Name	Value
is-content-area-overview	false
Content Areas	Knowledge/Coding Practices
SDLC Relevance	Implementation
Workflow State	Publishable